B. TECH. PROJECT REPORT

On

Common framework to enable businesses via Cloud using Open Source technologies

BY

DEEPIKA

DISCIPLINE OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

November 2017

Common framework to enable businesses via Cloud using Open Source technologies

A PROJECT REPORT
Submitted in partial fulfillment of the requirements for the award of the degrees
of
BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE AND ENGINEERING

Submitted by:
DEEPIKA

Guided by:
DR. GOURINATH BANDA
(Associate Professor, IIT INDORE)

INDIAN INSTITUTE OF TECHNOLOGY INDORE
November 2017

CANDIDATE’S DECLARATION

I hereby declare that the project entitled **“Common framework to enable businesses via cloud using open source technologies”** submitted in partial fulfillment for the award of the degree of Bachelor of Technology in ‘Computer Science and Engineering’ completed under the supervision of **Dr. Gourinath Banda, Associate Professor,** IIT Indore is an authentic work.

Further, we declare that I have not submitted this work for the award of any other degree elsewhere.

Deepika

(140001010)

CERTIFICATE by BTP Guide

It is certified that the above statement made by the student is correct to the best of my knowledge.

Dr. Gourinath Banda

Associate Professor

Department of Computer Science and Engineering

IIT Indore

Preface

This report on ”Common framework to enable businesses via cloud using open-source technologies” is prepared under the guidance of Dr. Gourinath Banda.

Through this report, I have tried to give a detailed description of Cloud and opensource software Openstack and its components. I have tried to show each and every step that are required to be followed to achieve my goal of providing an economically viable solution for business enterprises. Cost analysis for the same has been shown in this report. Benchmarks have been included to compare the performance of thin-client Raspberry Pi with the Ubuntu instance launched using openstack accessed by thin-client Raspberry Pi. Sceenshots of various openstack functionalities have been added to make the report more comprehensive and illustrative.
I have also mentioned the various challenges that I faced from time to time to get my job done and finally complete my project.

DEEPIKA

B. Tech, IV Year

Discipline of Computer Science and Engineering

IIT Indore

Acknowledgments

My efforts bore fruit with the successful completion of this project. I acknowledge the cooperation, encouragement and austerity of my project guide, Dr. Gourinath Banda, whose guidance did half the magic of keeping me thrilled throughout this project. As a student, I respected his authority, yet felt comfortable enough to share ideas, ask questions, and discuss. He is venerated for making me finish the jobs ahead of schedule. That's an environment that only few students get to truly experience.

However, there are many others who share the reward of this effort simply because it would never have been this good without their help. Having said that, I would like to express my heartfelt gratitude to Mr. Yogendra Singh who helped me in the area of networking.I would also like to thank my project partner Akshaya Kumar for his dedication towards the work. There were several points where his excellent problem-solving skills and creativity allowed us to swiftly resolve issues that might have easily escalated into roadblocks. He has done wonderful work in contributing his talents and skills to the project.

Deepika
B. Tech, IV Year
Discipline of Computer Science and Engineering
IIT Indore

Abstract

Cloud computing is a term that has gained widespread use over the last few years. To summarise this term, it's when you store data and programs on cloud instead of storing it on the hard drive of a computer. An internet connection is needed to access the data. It's about sharing, storing, processing and managing resources delivered over a network of remote servers.

With growing cloud computing trends, many companies are starting to use cloud for their business day by day. Small businesses and organizations are expected to buy new IT equipment. Whether they purchase a PC, notebook, server or other network hardware, they are likely to experience sticker shock once they factor in the total cost of ownership. But moving their business to the cloud can provide them with significant savings and hence achieve their break even point early.

Thin client/server computing is a server-based approach to delivering applications to end-users. The server-side infrastructure makes use of cloud computing such as application virtualization, hosted shared desktop (HSD) or desktop virtualization (VDI). A thin client is a lightweight computer built to connect to a server from a remote location. Nowadays Raspberry Pi thin client devices are viable for enterprises. Raspberry Pi is significantly cheaper than conventional desktop PC. End users (enterprise employees) can access full-fledged conventional desktop functionality via Raspberry Pi for much lower costs by using desktop virtualization aspect of cloud computing on the server-side of thin client/server model. Cloud computing fully utilizes hardware. Virtualization increases the value of physical server hardware, meaning businesses can do

more with less. As a result, small businesses will see a decrease in rack space, power usage, IT requirements, etc. That means lower installation, maintenance, hardware, upgrade and support costs. For small businesses, especially, these savings are invaluable.

OpenStack software is explored and used to set-up cloud and implement desktop virtualization. OpenStack is an open source Infrastructure as a Service (IaaS) initiative that controls large pools of compute, storage, and networking resources throughout a data center, managed through a dashboard.OpenStack is a set of software tools for building and managing cloud computing platforms for public and private clouds. Since some enterprises require their data to be highly secured and exist within the boundaries of their premises or country. This can be easily achieved by establishing an in-house cloud to keep critical data highly secured and no third party has access to your information.

Contents

1 Introduction 1

1.1 Problem Statement 1

1.2 Current Scenario 2

1.3 Proposed Solution 2

1.4 Usefulness of Project 2

1.5 Cost Analysis 3

2 The Cloud 5

2.1 Introduction 5

2.2 What is Cloud? 5

2.3 What is Cloud Computing? 5

2.4 Uses of Cloud Computing 6

2.5 Benefits of Cloud Computing 6

2.6 Forms of Cloud Computing 7

3 OpenStack 9

3.1 Introduction 9

3.2 OpenStack Components 9

3.3 Level Hierarchy of Hardware and Software 12

3.4 OpenStack Installation Procedure 12

3.5 Launching Instance in OpenStack from Dashboard 14

3.6 Configuring Cloud Instance 24

3.7 Configuring Thin Client Raspberry Pi 25

4 Challenges 29

5 Analysis **33**

6 What's Next? **35**

References **37**

List of Figures

Figure 3.1: OpenStack Components

Figure 3.2: Level Hierarchy

Figure 3.3: OpenStack Dashboard

Figure 3.4: Projects in OpenStack

Figure 3.5: Volumes in OpenStack

Figure 3.6: Network Topology in OpenStack

Figure 3.7: Instances in OpenStack

Figure 3.8: Login Screen of Remmina Remote Desktop Client

Figure 3.9: Login Screen of xrdp

Figure 3.10: Login process of xrdp

Figure 3.11: xfce Desktop Environment

Figure 3.12: Homepage of Django website

Figure 4.1: Unsuccessful Ping from instances to 8.8.8.8 which is a static assigned Corporate IP address allocated to Google

Figure 4.2: Hardware Setup I

Figure 4.3: Hardware Setup II

Figure 5.1: Performance comparison between Raspberry Pi and Cloud Instance

1

Introduction

1. Enterprises from small to large require automation. To achieve that, software applications are required which further need hardware. Hardware is expensive and needs power for its operation. Paradigm of cloud computing could cut monetary expenses and energy consumption.

2. Digitization is very important in India. Digitizing information makes it easier to preserve, access and share. Businesses are transforming and creating new services by rewiring and integrating existing business processes. Internet ,as is obvious, is the platform that bridges the divide between rural and urban India. With the availability of cellular data even in far reached corners of the country we can provide even small scale industries and local businesses with the opportunity to manage its resources and bookkeeping. On the other hand with the scalability available on the cloud we can serve large industries as easily as acquiring new hardware.

3. New technological advancements have been made for reduction of cost and power consumption using ultra thin clients like Raspberry Pi.

1.1 Problem Statement

The problem statement is to create a common framework to enable businesses via cloud using opensource technologies.

1.2 Current Scenario

New enterprises often face scarcity of funds so they would have to invest in hardware for longer duration . Acquiring new hardware in bulk is difficult for a new institute, also space and transportation is required. Configuration of every desktop with standard softwares that are required takes a lot of time and manpower.

1.3 Proposed Solution

- We shift all computing power to an in-house cloud that can be scaled as required.
- We then provide access to the instances of a personal desktop created on the cloud and this saves us time for configuration of every desktop as it can be done via a standard image we create beforehand.
- These instances are accessed by thin clients like Raspberry Pi via SSH or RDP and this saves us the cost of acquiring new desktops in bulk.
- We dynamically manage these instances to allocate and free space in the cloud. So we can always have ability to serve the clients and also if needed we can easily scale up the cloud.
- With this we can assure that the institute can reach break even point sooner than before.

1.4 Usefulness of Project

We are creating a framework for a business enterprise to reduce their hardware and software related costs.
We assume that institute has

- 200 employee.

1.5 Cost Analysis

If each staff member has their own personal system then this will cost the institute as below.
We estimate the cost using Dell Inspiron 3268 Small Desktop with the specifications mentioned below:

- Intel Pentium G4560 Processor (3MB Cache up to 3.50 GHz)
- Windows 10 Home 64-bit English
- 4GB, DDR4, 2400MHz
- 1TB 7200 rpm Hard Drive

Price of each unit=29,790 INR(*as given on dell.com)
Total cost for 200 desktops =59,58,000 INR
In our solution we suggest a server with the following configuration:

- Cores: 16
- RAM: 128 GB

Cost of each server = 3,04,300 INR
By using Opensource technologies, we try to reduce software related costs. We would reduce power consumption and space requirements by utilizing ultra thin client base and thus we can get a new organization that could steal a march on rivals by minimizing the lead time minimization and minimal usage of resources.
OpenStack virtualization gives us an opportunity to over-commit, hence we are able to create instances which can use resources more than the actual hardware can provide.
OpenStack scheduler has the following over-committing ratios:

- Cores: 16:1
- RAM: 1.5:1

Therefore total virtual cores can be assigned by server =16*16= 256
Assuming each instance has 4 cores and 4 GB RAM which is same as the desktop from dell.
We can have number of instance =128*1.5/4 =48 instances running on one server
Number of cores that are required = 48*4 =192 (We have 256 virtual cores available) Cost of thin client (Raspberry Pi with display) (price approximated from amazon):

- 3812 INR raspberry complete kit
- 5000 INR for 15" monitor
- 1000 INR for keyboard+mouse+hdmi to vga converter

Total = 9,812 INR for single thin client
Total for all 48 clients = 4,70,976 INR
Total cost of server + 48 clients= 4,70,976 INR + 3,04,300 INR = 7,75,276 INR
If 48 desktop were bought = 14,29,920 INR
Cost difference = 6,54,644 INR
Therefore total savings for server 200 clients = 27,27,983 INR
Furthermore, we assumed that all the clients will access their instances simultaneously but that is never the case. We have planned the dynamic assignment of instances on "On Demand" basis. This reduces the actual requirement of number of instances. Our solution also reduces the power requirement and power back up can support the thin client for longer duration during power cuts.

In next chapter, we will have quick introduction to cloud which forms the basis of implementation for our solution.

2

The Cloud

2.1 Introduction

With advancement in Distributed computing, the limits of computational power and storage capacity are not dependent on any individual machine. Along with the development of virtualization we can set a single virtual machine even with not all hardware being present on a single geographical location. Cloud computing is versatile, it can be configured to serve tasks requiring high computational capacity or with jobs requiring large amounts of data to be stored. It can be configured to serve a small set of people in a few hundreds or an institution involving millions of clients, just by addition of new hardware.

2.2 What is Cloud?

It's somewhere at the other end of your internet connection – a place where you can access apps and services, and where your data can be stored securely.

2.3 What is Cloud Computing?

Cloud computing is the delivery of computing services like servers, storage, databases, networking, software, analytics and more—over the Internet ("the cloud").

2.4 Uses of Cloud Computing

The first cloud computing services are barely a decade old, but already a variety of organizations from tiny start-ups to global corporations, government agencies to non–profits are embracing the technology for all sorts of reasons. Here are a few of the things you can do with the cloud:

- Create new apps and services
- Store, backup and recover data
- Host websites and blogs
- Stream audio and video
- Deliver software on demand
- Analyze data for patterns and make predictions

2.5 Benefits of Cloud Computing

Cost Cloud computing eliminates the capital expense of buying hardware and software and setting up and running on-site data centers(data center refers to on-premise hardware that stores data within an organization's local network)—the racks of servers, the round-the-clock electricity for power and cooling, the IT experts for managing the infrastructure. It adds up fast.

Speed Most cloud computing services are provided with self service and on demand, so even vast amounts of computing resources can be provisioned in minutes, typically with just a few mouse clicks, giving businesses a lot of flexibility and taking the pressure off capacity planning.

Global Scale The benefits of cloud computing services include the ability to scale elastically. In cloud speak, that means delivering the right amount of IT resources-for example, more or less computing power, storage, bandwidth-right when it's needed and from the right geographic location.

Productivity On-site data centers typically require a lot of “racking and stacking”-hardware setup, software patching and other time-consuming IT management chores. Cloud computing removes the need for many of these tasks, so IT teams can spend time on achieving more important business goals.

Performance The biggest cloud computing services run on a worldwide network of secure data centers, which are regularly upgraded to the latest generation of fast and efficient computing hardware. This offers several benefits over a single corporate data center, including reduced network latency for applications and greater economies of scale.

Reliability Cloud computing makes data backup, disaster recovery and business continuity easier and less expensive, because data can be mirrored at multiple redundant sites on the cloud provider’s network.

2.6 Forms of Cloud Computing

HaaS(Hardware as a Service) In the Hardware-as-a-Service model, hardware that belongs to a managed service provider (MSP) is installed at a customer’s site and a service level agreement (SLA) defines the responsibilities of both parties

IaaS(Infrastructure as a Service) With IaaS, you rent IT infrastructure-servers and virtual machines (VMs), storage, networks, operating systems from a cloud provider.

Paas(Platform as a Service) Platform-as-a-Service (PaaS) refers to cloud computing services that supply an on-demand environment for developing, testing, delivering and managing software applications. PaaS is designed to make it easier for developers to quickly create web or mobile apps, without worrying about setting up or managing the underlying infrastructure of servers, storage, network and databases needed for development.

SaaS(Software as a Service) Software-as-a-Service (SaaS) is a method for delivering software applications over the Internet. With SaaS, cloud providers host and manage the software application and underlying infrastructure and handle any maintenance, like software upgrades and security patching.

In next chapter, we will introduce OpenStack and its components.

3

OpenStack

3.1 Introduction

OpenStack is an open source Infrastructure as a Service (IaaS) initiative that controls large pools of compute, storage, and networking resources throughout a datacenter, managed through a dashboard.OpenStack is a set of software tools for building and managing cloud computing platforms for public and private clouds.

3.2 OpenStack Components

Please refer Fig 3.1 for various openstack components.

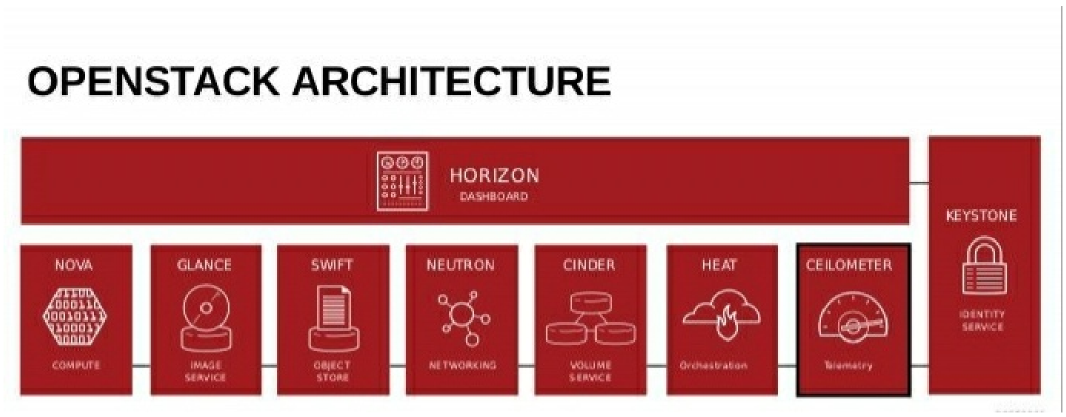


Figure 3.1: OpenStack Components

Keystone Keystone is an OpenStack service that provides API client authentication, service discovery, and distributed multi-tenant authorization by implementing OpenStack's Identity API.

Authentication The process of confirming the identity of a user. To confirm an incoming request, OpenStack Identity validates a set of credentials users supply.

Tenant A tenant also known as project. It has resources such as users,images,instances, networks and security groups. These resources are only visible to that particular project. A user can belong to one or more tenants and is able to switch between these projects to gain access to those resources.
In simple words, it authorizes users, services and endpoints. Keystone uses tokens for authorization and maintains Session state.

Glance The Image service (glance) project provides a service where users can upload and discover data assets that are meant to be used with other services. This currently includes images and metadata definitions.Glance image services include discovering, registering, and retrieving virtual machine (VM) images.

Metadata Metadata is data that describes other data. Metadata summarizes basic information about data, which can make finding and working with particular instances of data easier. For example, author, date created and date modified and file size are examples of very basic document metadata. Having the ability to filter through that metadata makes it much easier for someone to locate a specific document.
In simple words glance is the Image Registry, it stores and Manage our guest (VM) images, Disk Images, snapshots etc. It also contains pre-built VM templates so that you can try it on the fly. Instances are booted from our glance image registry. User can create custom images and upload them to Glance later reuse. A feature of Glance is to store images remotely so to save local disk space.

Neutron Neutron, the networking component which provides the software defined Networking Stack for OpenStack. It Provides Networking as a Service. It gives the cloud tenants an API to build rich networking topologies, and configure advanced network policies in the cloud. It enables innovation plugins (open and closed source) that introduce advanced network capabilities which let anyone build advanced network services (open and closed source) that plug into OpenStack tenant networks means that you can create advanced managed switches and routers. You can even create an intelligent switch from a PC (Yes you can use it as a standalone component) and use it to replace your managed switch or at least make it act as a backup Switch.

Nova Nova compute or the king service provides a platform on which we are going to run our guest machines; It's the virtual machine provisioning and management module that defines drivers that interact with underlying virtualization. It provides a Control plane for an underlying hypervisors. Each hypervisor requires a separate Nova Instance. Nova supports almost all hypervisors known to man.
It requires the following additional OpenStack services for basic function:

- Keystone: This provides identity and authentication for all OpenStack services.
- Glance: This provides the compute image repository. All compute instances launch from glance images.
- Neutron: This is responsible for provisioning the virtual or physical networks that compute instances connect to on boot.

Horizon Horizon is the canonical implementation of OpenStack's Dashboard, which provides a web based user interface to OpenStack services including Nova, Keystone, etc.

3.3 Level Hierarchy of Hardware and Software

Level Hierarchy is used as shown in Fig 3.2.

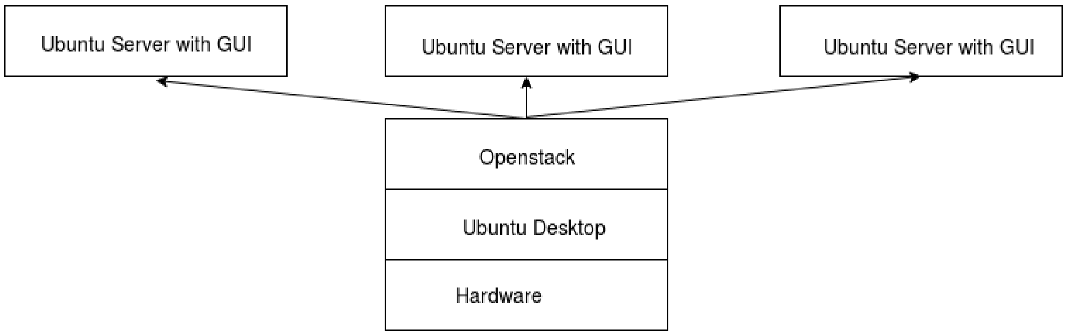


Figure 3.2: Level Hierarchy

3.4 OpenStack Installation Procedure

The requirements

- An updated Ubuntu 16.04 LTS server/desktop
- 14 GB of RAM (minimum)
- 100 GB of hard disk space (minimum)

Update and install dependencies First, you must update Ubuntu. Open a terminal window and issue the commands:

```
sudo apt-get update
sudo apt-get upgrade
```

Once those commands have completed, you'll need to install git. To do this, issue the command:

```
sudo apt-get install git
```

Allow that installation to complete.

Clone devstack We're going to use git to clone devstack. To do this, go back to your terminal window and issue the following two commands:

```
cd /
sudo git clone https://git.openstack.org/openstack-dev/devstack master
```

Next, we have to copy the sample configuration file and set a password to be used for automated deployment. To complete this task, issue the following commands:

```
cd devstack
sudo cp samples/local.conf local.conf
```

Now we must set the automated deployment password. Open the local.conf file with the command:

```
sudo nano local.conf
```

Search for the password variable section and ensure it reflects the following (YOURPASSWORD is the actual password you want to use):

```
ADMIN_PASSWORD=YOURPASSWORD
MYSQL_PASSWORD=$ADMIN_PASSWORD
RABBIT_PASSWORD=$ADMIN_PASSWORD
SERVICE_PASSWORD=$ADMIN_PASSWORD
```

We need to manually set the network interface as follows:

```
FLAT_INTERFACE=eth0
```

Here eth0 is the network interface in ifconfig of host machine.

The FLAT_INTERFACE value indicates the network interface card openstack will use for network access.

Create a user and start the installation It's time to run a few scripts. The first script will create a new user for devstack. The command to run this script is:

```
sudo /devstack/tools/create-stack-user.sh
```

Once the script completes, you'll need to change the permissions of the devstack folder with the command:

```
sudo chown -R stack:stack /devstack
```

It's time to run the installation script-this must be run as the stack user by first issuing the command sudo su stack. After you change to the stack user, kick off the install with the command /devstack/stack.sh. This command will take at least 30 minutes to complete.

When the command finally completes, you will be given an IP address (the address of the server), as well as two usernames (the admin password was created in the local.conf file). Open a browser, point it to http://SERVER_IP_ADDRESS/dashboard, and log in with the given credentials.

3.5 Launching Instance in OpenStack from Dashboard

Configure access and security for instances

Before you launch an instance, you should add security group rules to enable users to ping and use SSH to connect to the instance. Security groups are sets of IP filter rules that define networking access and are applied to all instances within a project. To do so, you either add rules to the default security group Add a rule to the default security group or add a new security group with rules. Key pairs are SSH credentials that are injected into an instance when it is launched. To use key pair injection, the image that the instance is based on must contain the cloud-init package. Each project should have at least one key pair.
When an instance is created in OpenStack, it is automatically assigned a fixed IP address in the network to which the instance is assigned. This IP address is permanently associated with the instance until the instance is terminated. However, in addition to the fixed IP address, a floating IP address can also be attached to an instance. Unlike fixed IP addresses, floating IP addresses are able to have their associations modified at any time, regardless of the state of the instances involved.

Add a rule to the default security group

This procedure enables SSH and ICMP (ping) access to instances. The rules apply to all instances within a given project, and should be set for every project unless there is a reason to prohibit SSH or ICMP access to the instances.
This procedure can be adjusted as necessary to add additional security group rules to a project, if your cloud requires them. When adding a rule, you must specify the protocol used with the destination port or source port.

1. Log in to the dashboard as shown in Fig 3.3.

2. Select the appropriate project from the drop down menu at the top left. Various projects in openstack are shown in Fig 3.4.

3. On the Project tab, open the Compute tab and click Access & Security category. The Security Groups tab shows the security groups that are available for this project.

4. Select the default security group and click Manage Rules.

5. To allow SSH access, click Add Rule.

6. In the Add Rule dialog box, enter the following values:
   Rule: SSH
   Remote: CIDR
   CIDR: 0.0.0.0/0

7. Click Add.

8. Instances will now have SSH port 22 open for requests from any IP address.

9. To add an ICMP rule, click Add Rule.

10. In the Add Rule dialog box, enter the following values: Rule: All ICMP
    Direction: Ingress
    Remote: CIDR
    CIDR: 0.0.0.0/0

11. Click Add.

12. Instances will now accept all incoming ICMP packets.

Extend Cinder-Volume

Volume Groups of OpenStack has size of 10 GB by default. This implies we can't make volumes of size greater than 10 GB and hence can't launch instance of having disk size greater than 10 GB. So we require to extend the size of

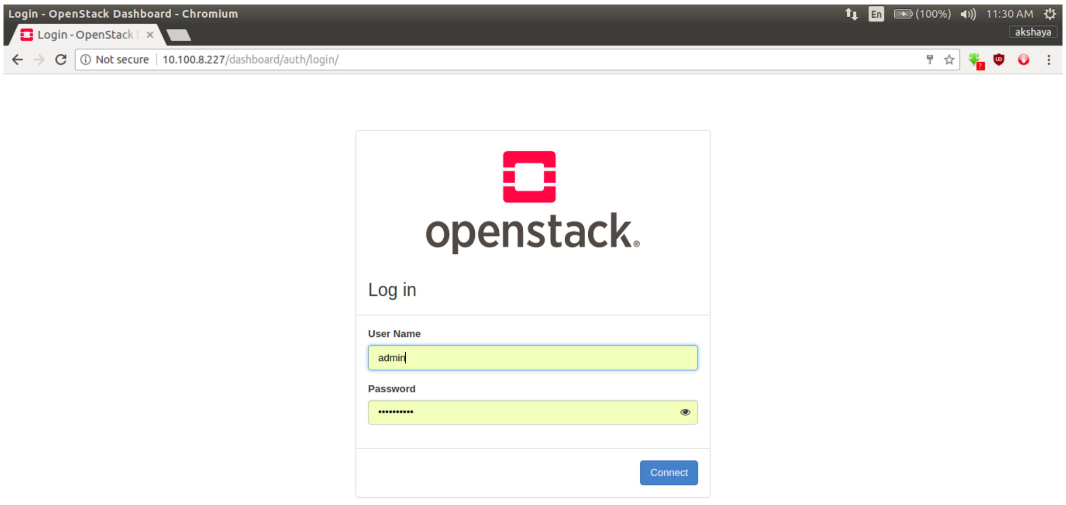
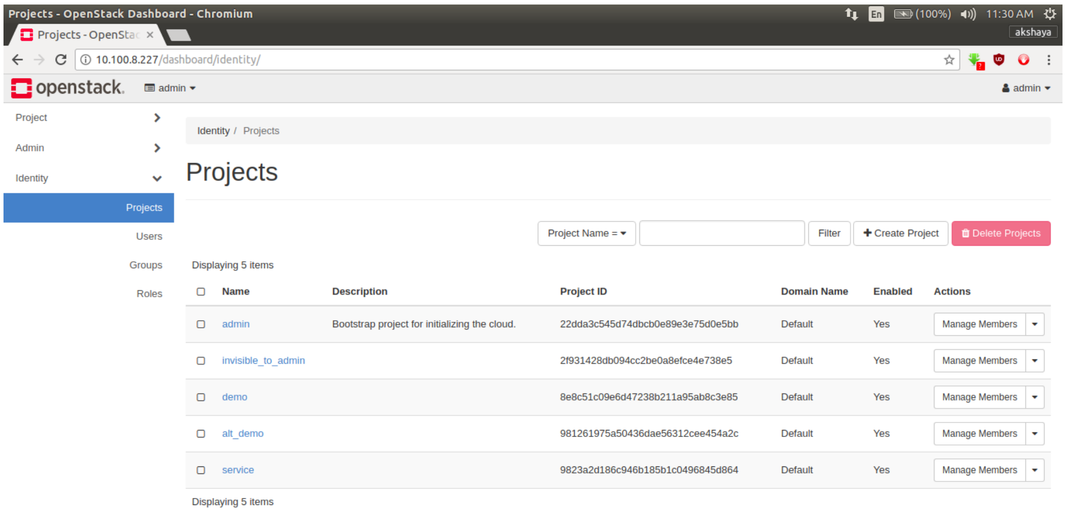


Figure 3.3: OpenStack Dashboard



Projects

Displaying 5 items

Name	Description	Project ID	Domain Name	Enabled	Actions
admin	Bootstrap project for initializing the cloud.	22dda3c545d74dbcb0e89e3e75d0e5bb	Default	Yes	Manage Members
invisible_to_admin		2f931428db094cc2be0a8efce4e738e5	Default	Yes	Manage Members
demo		8e8c51c09e6d47238b211a95ab8c3e85	Default	Yes	Manage Members
alt_demo		981261975a50436dae56312cee454a2c	Default	Yes	Manage Members
service		9823a2d186c946b185b1c0496845d864	Default	Yes	Manage Members

Displaying 5 items

Figure 3.4: Projects in OpenStack

volume groups. Execute vgs command to see the volume groups of the server. This should be the volume group (stack-volumes) OpenStack uses to create

```
$vgs
  VG              #PV #LV #SN Attr   VSize   VFree
  stack-volumes     3   1   0 wz--n-  10.00g 10.00g
  stratos1          1   2   0 wz--n- 931.09g 48.00m
```

volume and volume snapshots. In order to create more volumes and larger volumes you have to increase the capacity of the volume group. In this case the volume group to extend is ”stack-volumes”. In our case the volume group to extend is ”stack-volumes-lvmdriver-1”. So we do the changes accordingly.

Create a partition with 50 GB or whatever you like for all your instances. Your instances are going to use this partition for making their disk volumes.

```
dd if=/dev/zero of=cinder-volumes bs=1 count=0 seek=50G
losetup /dev/loop3 cinder-volumes
fdisk /dev/loop3
```

And at the fdisk prompt, enter the following commands:

```
n
p
1
ENTER
ENTER
t
8e
w
```

Create a physical volume:

```
root@stratos1:~# pvcreate /dev/loop3
Physical volume ”/dev/loop3” successfully created
```

Extend the volume group ”stack-volumes”:

```
root@stratos1:~# vgextend stack-volumes /dev/loop3
Volume group ”stack-volumes” successfully extended
```

Creating Image

We have to create volume of Ubuntu server and for that we need to download the cloud image for the same which you can get from here https://cloud-images.ubuntu.com/ in .img format. Once we have downloaded the image we are ready to create image as follows:

1. Login to the dashboard.

2. Select the Project option on the left and the click on the Compute tab and under that go to Images.

3. Next click on Create Image option and a dialog box pops up with several options

4. In Image Name tab, assign a name to the volume. Image Description tab is up to you.

5. Then comes Image Source section, in that Browse the file in your system. Here you need to know the format of the file and for that go to the terminal and give command analogous to this file /home/user/Downloads/xenial-server-cloudimg-amd64-disk1.img And you will get Format for the same and set the Format dropdown to the same. Then comes, Image Sharing section, Set Visibility to public and Protected to No and after this go ahead click on Create Image.

Creating Volume

1. Login to the dashboard.

2. Select the Project option on the left and the click on the Volumes tab and under that go to Volumes.

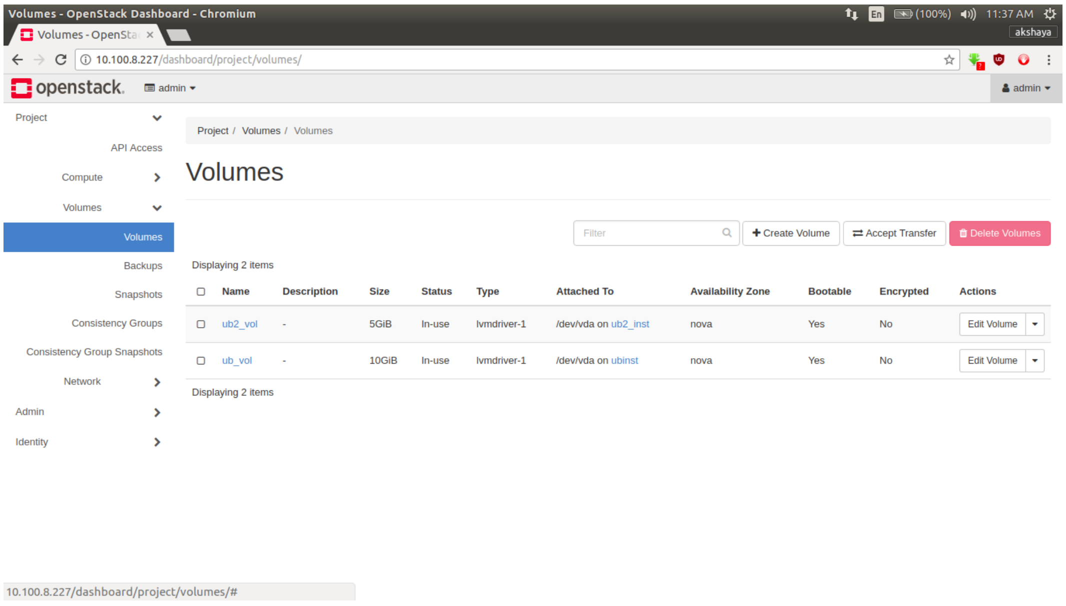
3. Next click on Create Volume option and a dialog box pops up with several options.

4. In Volume Name tab, assign a name to the volume. Description tab is up to you.

5. Then comes Volume Source dropdown, select Image option.

6. After the Use Image as a source dropdown appears and from there choose your image. Then comes Type tab, no need to make alterations in that. Size tab is there, assign size as required and set the Availability Zone tab value to nova, if not set.

7. Go ahead and hit Create Volume button in blue color and you have created the volume as shown in Fig 3.5.



Project / Volumes / Volumes

Volumes

Displaying 2 items

Name	Description	Size	Status	Type	Attached To	Availability Zone	Bootable	Encrypted	Actions
ub2_vol	-	5GiB	In-use	lvmdriver-1	/dev/vda on ub2_inst	nova	Yes	No	Edit Volume
ub_vol	-	10GiB	In-use	lvmdriver-1	/dev/vda on ubinst	nova	Yes	No	Edit Volume

Displaying 2 items

Figure 3.5: Volumes in OpenStack

Create Network

We here create a network(subnet) for our instance.

1. Log in to the dashboard.
2. Select the appropriate project from the drop down menu at the top left.
3. On the Project tab, open the Network tab and click Networks category.
4. Click Create Network.In the Create Network dialog box, assign a name to the network in Network Name and then following 3 checkboxes. Check Enable Admin State and Create Subnet checkboxes and hit Next.
5. In Subnet Name specify a name for the subnet.
6. In Network Address Source dropdown, go for Enter Network Address Manually.
7. In Network Address tab, fill this value 192.168.1.0/24 (this value can vary for different subnets like it could also be 192.168.2.0/24. This implies that instances having this 192.168.1.0/24 subnet and this 192.168.2.0/24 subnet belongs to different subnets and in order to connect them we need router which we will discuss later).
8. IP Version: Select IPv4 or IPv6.

9. Gateway IP: This parameter is optional so we leave it as it is.

10. Disable Gateway: Don't check this check box and hit Next.

11. Enable DHCP: Select this check box to enable DHCP and hit Create.

12. The dashboard shows the network on the Networks tab.

Launch and manage instances

1. Log in to the dashboard.

2. Select the appropriate project from the drop down menu at the top left.

3. On the Project tab, open the Compute tab and click Instances category.

4. The dashboard shows the instances with its name, its private and floating IP addresses, size, status, task, power state, and so on.

5. Click Launch Instance.

6. In the Launch Instance dialog box, we specify only the required fields to fulfill our purpose.

7. Instance Name:Assign a name to the virtual machine.

8. Availability Zone: Nova and Count: 1 and hit Next.

9. By default, this value is set to the availability zone given by the cloud provider (for example, us-west or apac-south). For some cases, it could be nova.

10. In Instance Boot Source, select Volume and Set the value to Yes for Delete Volume on Instance Delete.

11. Volume you created would be shown in Available section, move that volume to Allocated section using upward arrow and hit Next.

12. Now Specify the size of the instance to launch, according to the size of the volume you created and hit Next.

13. Now allocate the Network to your instance. Network you created earlier would be shown in Available section and move that to Allocated section and hit Next.

14. Don't modify Network Ports and hit Next. Then comes Security Groups, default value is allocated there so hit Next.

15. Now comes Key Pair, as of now we haven't created any key pair so go ahead click on Create Key Pair and dialogue box appears. Specify Key Pair Name and click on Copy Private Key to Clipboard and then save the key pair somewhere with you as we are going to use this key pair to ssh to the instance. Click on Create Keypair and hit Next.

16. I am going to insert a password while booting the instance in Customization Script as
```
#cloud-config
password: mypasswd
chpasswd:  expire: False
ssh_pwauth: True
```

17. Now mypasswd is the password to login into my Ubuntu server with default username as Ubuntu. Now click on Launch Instance.

18. The instance starts on a compute node in the cloud.

19. Now if you go to Network Topology as shown in Fig 3.6 and click on Graph you will see that we have a subnet with instance in it and it is not connected to the outside world. So we need a router to connect it to the outside world.

Create a router

1. Log in to the dashboard.

2. Select the appropriate project from the drop down menu at the top left.

3. On the Project tab, open the Network tab and click Routers category. Click Create Router.

Topology.png

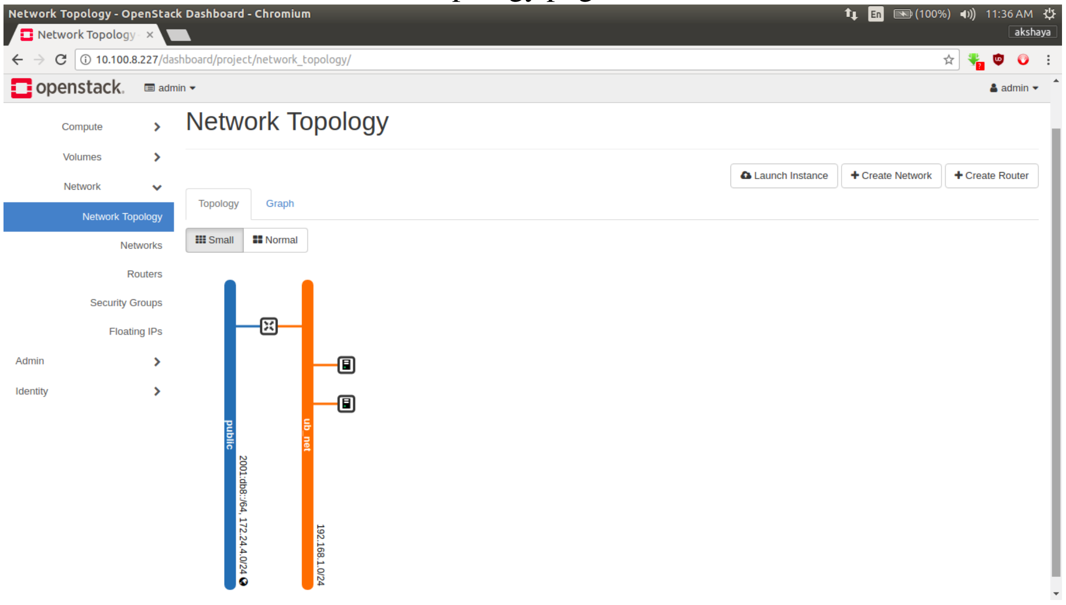


Figure 3.6: Network Topology in OpenStack

4. In the Create Router dialog box, specify a name for the router and External Network, and click Create Router.

5. The new router is now displayed in the Routers tab.

6. To connect a private network to the newly created router, perform the following steps:

   (a) On the Routers tab, click the name of the router.

   (b) On the Router Details page, click the Interfaces tab, then click Add Interface.

   (c) In the Add Interface dialog box, select a Subnet.

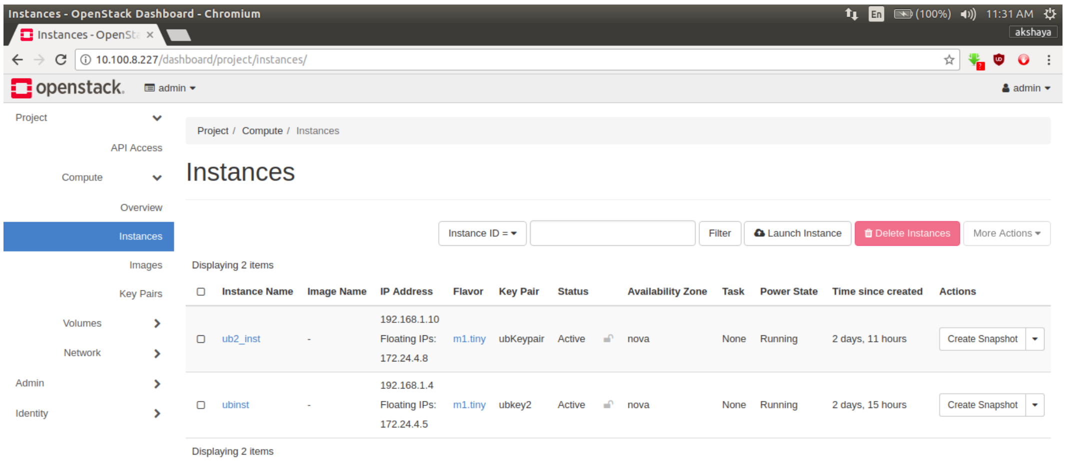
   (d) Optionally, in the Add Interface dialog box, set an IP Address for the router interface for the selected subnet.

   (e) If you choose not to set the IP Address value, then by default OpenStack Networking uses the first host IP address in the subnet.

   (f) The Router Name and Router ID fields are automatically updated.

7. Click Add Interface.

You have successfully created the router. You can view the new topology from the Network Topology tab. We add public interface as well. Now we need an IP for public connection so we associate Floating IP to our instance.


Instances - OpenStack Dashboard - Chromium

10.100.8.227/dashboard/project/instances/

Project / Compute / Instances

Instances

Displaying 2 items

Instance Name	Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
ub2_inst	-	192.168.1.10 Floating IPs: 172.24.4.8	m1.tiny	ubKeypair	Active	nova	None	Running	2 days, 11 hours	Create Snapshot
ubinst	-	192.168.1.4 Floating IPs: 172.24.4.5	m1.tiny	ubkey2	Active	nova	None	Running	2 days, 15 hours	Create Snapshot

Displaying 2 items

Figure 3.7: Instances in OpenStack

Associate Floating IP

1. Log in to the dashboard.
2. Select the appropriate project from the drop down menu at the top left.
3. On the Project tab, open the Compute tab and click Instances category as shown in Fig 3.7.
4. Go to Actions column and click on Associate Floating IP in dropdown and dialog box appears and get an IP from public pool.

We can boot our instance. From the same above described Actions Column select Console from the dropdown and once instance finishes booting you can login with username and password to the instance.

3.6 Configuring Cloud Instance

We launched the cloud instances with the below configurations by following the steps mentioned in above section:

1. RAM = 4 GB
2. Disk size = 50 GB
3. Ubuntu server = 14.04 LTS (Trusty Tahr)

Install GUI on the instance

For our purpose, we make instance by using cloud image for Ubuntu server 12.04. Here we are installing GUI for the same. Open the console of instance and login with default username: ubuntu and password as mentioned in configuration script during the launch of instance.
We need to resolve hostname. Give the command sudo nano /etc/hosts and add 127.0.0.1 name_of_your_machine and save the changes.
We have to resolve dns server. Give the command sudo nano /etc/resolv.conf and add nameserver your_dns_server and save the changes.
Update the instance by hitting sudo apt-get update.
Upgrade the instance by giving sudo apt-get upgrade command.
Install the GUI in the Ubuntu server by using the command sudo apt-get install ubuntu-gnome-desktop. It will take a while. Once it finishes installation, hit sudo startx and we will get a full-fledged gnome desktop.

Start XRDP on the instance

In the terminal, type the following command:
sudo apt-get install xrdp
Install the xfce4 Desktop Environment by using following commands:
sudo apt-get update
sudo apt-get install xfce4
Configure xrdp to use xfce desktop environment:
echo xfce4-session ≻ ∼/.xsession

Restart the xrdp service by issuing the following command:

```
sudo service xrdp restart
```

3.7 Configuring Thin Client Raspberry Pi

Once the Cloud Instance is configured, we configure the Raspberry Pi to access the Cloud Instance via SSH or RDP.
The thin client i.e Raspberry Pi we are using for our project is having following configurations:

- RAM = 1 GB
- Disk Size = 8 GB

Enable SSH

We hit raspi-config in the terminal:

1. Enter sudo raspi-config in a terminal window.
2. Select Interfacing Options.
3. Navigate to and select SSH.
4. Choose Yes.
5. Select Ok.
6. Choose Finish.

Alternatively, we can use systemctl to start the service:

```
sudo systemctl enable ssh
sudo systemctl start ssh
```

Connect to cloud instance by using SSH via Raspberry Pi

Enable the SSH as described above and then do the following for connection:
To use SSH to connect to your instance, use the downloaded keypair file. Note:The username is ubuntu for the Ubuntu cloud images on TryStack.

1. Copy the IP address for your instance.

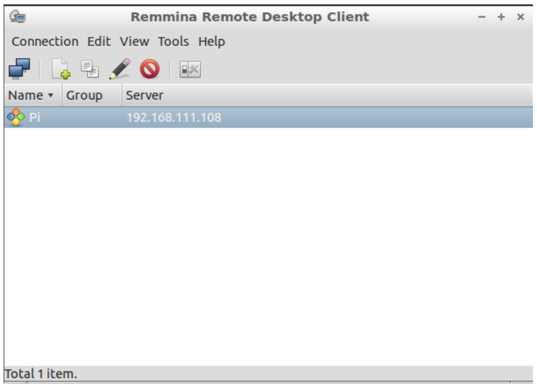
2. Use the ssh command to make a secure connection to the instance. For example:

3. $ ssh -i MyKey.pem ubuntu@10.0.0.2 (Use floating ip associated with the instance)

4. At the prompt, type yes.

Connect to cloud instance by using XRDP via Raspberry Pi

Install Remmina Remote Desktop Client in Raspberry Pi by using following command: sudo apt-get install remmina remmina-plugin-rdp
Use the floating ip address associated with the instance.
Now start remote Desktop client and enter the ip address as shown in Fig 3.8.
You should see then the login screen of xrdp presented to you in Fig 3.9. Note

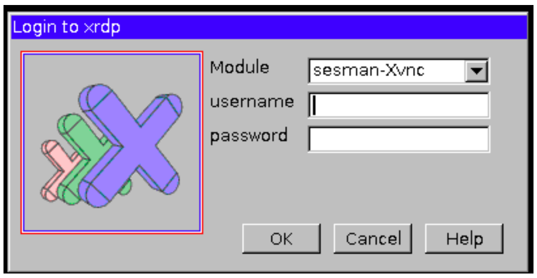

Remmina Remote Desktop Client
Connection Edit View Tools Help

Name	Group	Server
Pi		192.168.111.108

Total 1 item.

Figure 3.8: Login Screen of Remmina Remote Desktop Client

that, at this screen (and because we have not configured keyboard layout yet), the keyboard layout is set to English by default.
Enter your username and password and Press OK. You will see a dialog box


Login to xrdp

Module: sesman-Xvnc
username:
password:

Figure 3.9: Login Screen of xrdp

showing the login process as shown in Fig 3.10.

If everything is configured correctly,you should see your xfce desktop loading

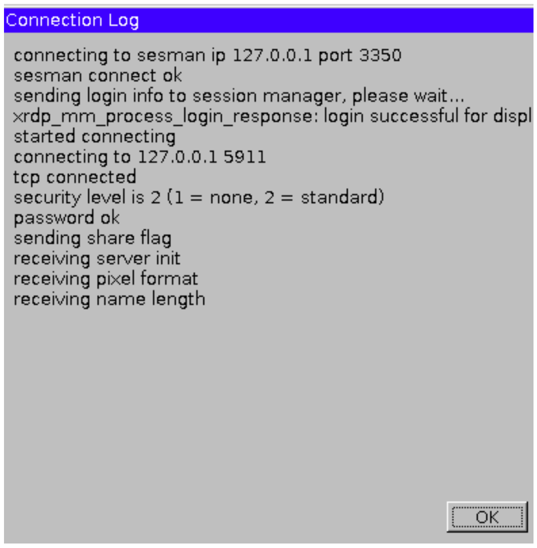

```
connecting to sesman ip 127.0.0.1 port 3350
sesman connect ok
sending login info to session manager, please wait...
xrdp_mm_process_login_response: login successful for displ
started connecting
connecting to 127.0.0.1 5911
tcp connected
security level is 2 (1 = none, 2 = standard)
password ok
sending share flag
receiving server init
receiving pixel format
receiving name length
```

Figure 3.10: Login process of xrdp

and you should be able to perform you work through this desktop environment as shown in Fig 3.11.

”On-Demand” Allocation of Resources

In order to provide the feature of dynamic allocation of resources (RAM) we have made a website by making use of the libraries provided by OpenStack SDK to work with OpenStack compute service along with Django and Python. Django (is an open-source web framework, written in Python, which follows the model-view-template (MVT) architectural pattern.
The website homepage has a login portal as shown in Fig 3.12. The username for this login portal is the same as the name of instance assigned to the employee. Once the employee successfully logs in into the website, he/she can now pause or resume the instance assigned to him/her.

This website also has a webpage for administrative functionalities. A pre-authorized user can add users for the above described functionality via this admin page. For adding a new user, first openstack instance has to be created and then a user via the admin page should be created for the same instance.
In next chapter, we will mention the challenges that we faced during our project.

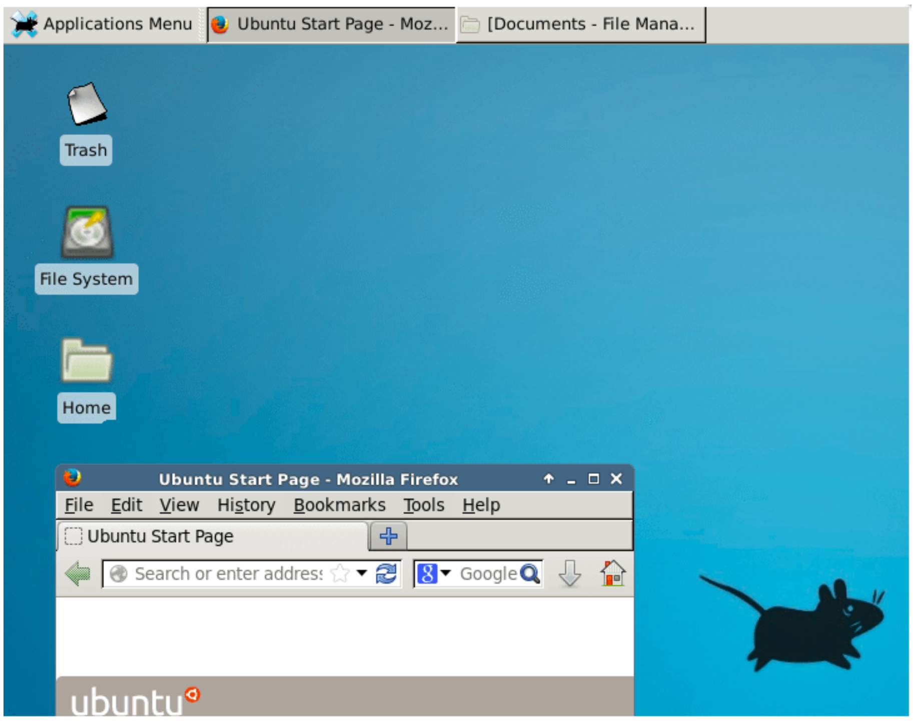

Applications Menu
Ubuntu Start Page - Moz...
[Documents - File Mana...
Trash
File System
Home
Ubuntu Start Page - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Ubuntu Start Page
Search or enter address
Google
ubuntu

Figure 3.11: xfce Desktop Environment

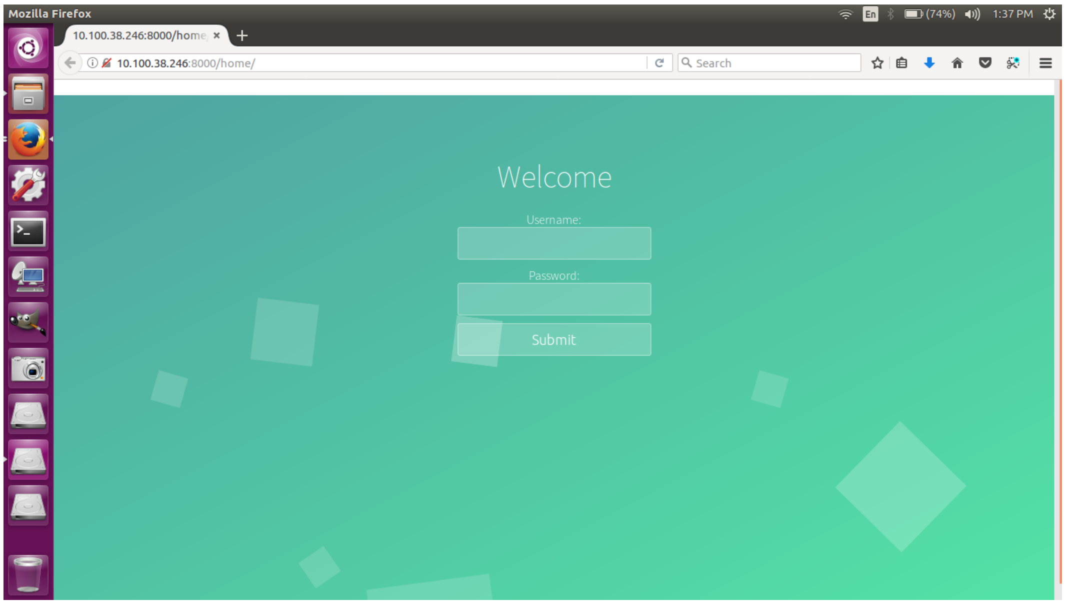


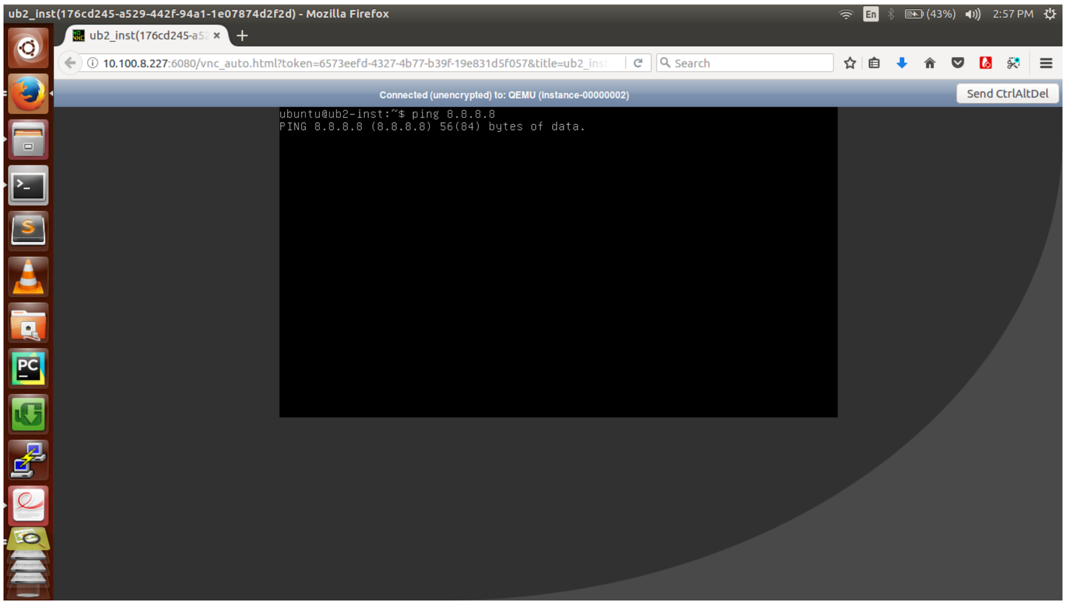
Figure 3.12: Homepage of Django website

4

Challenges

We faced many challenges on our way to the completion of this project.
After launching the instance of Ubuntu server in openstack, we couldn't login into the Ubuntu instance and were unable to ssh into our instance as there was no internet connectivity in the instance as shown in the figure. In order to get the network configurations right, we need to have access to the terminal. So for that configuration script was made to set password for the user in instance (as discussed earlier in section 3.5).

We used two hardware setups for our project as shown in Fig 4.2 and Fig 4.3.


ub2_inst(176cd245-a529-442f-94a1-1e07874d2f2d) - Mozilla Firefox
ub2_inst(176cd245-a5
10.100.8.227:6080/vnc_auto.html?token=6573eefd-4327-4b77-b39f-19e831d5f057&title=ub2_inst
Search
Connected (unencrypted) to: QEMU (instance-00000002)
Send CtrlAltDel
2:57 PM

```
ubuntu@ub2-inst:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
```

Figure 4.1: Unsuccessful Ping from instances to 8.8.8.8 which is a static assigned Corporate IP address allocated to Google

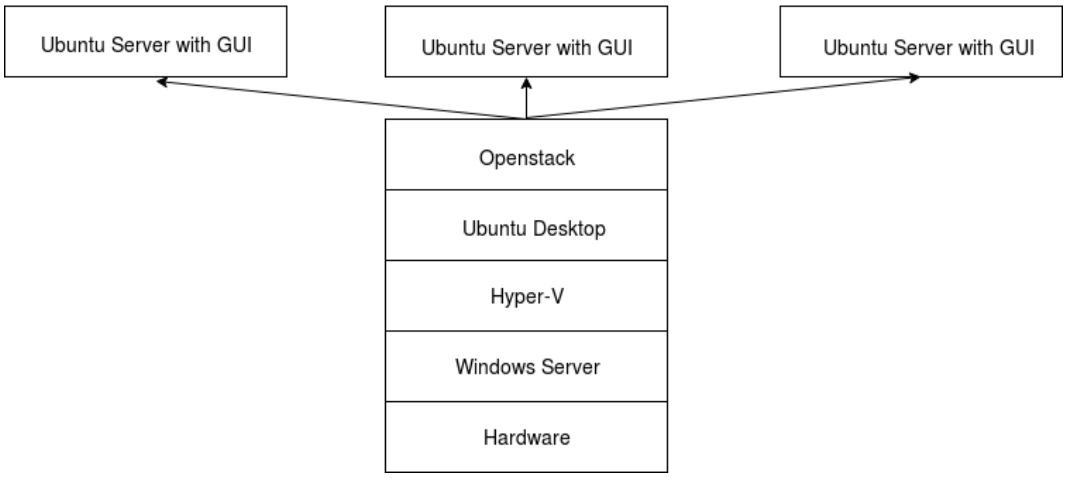


Figure 4.2: Hardware Setup I

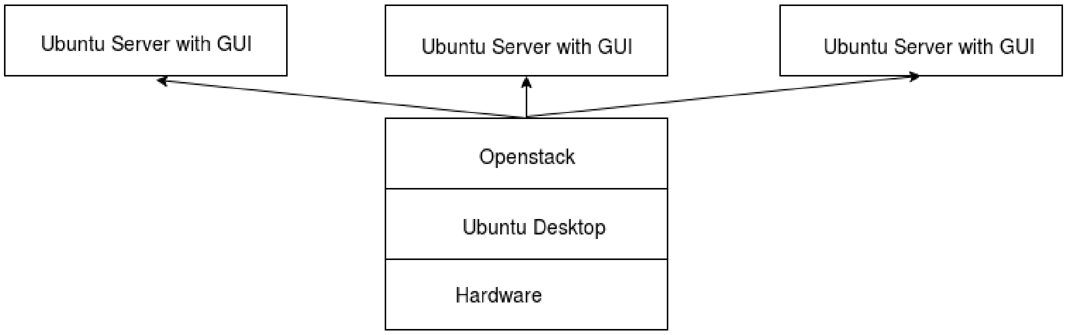


Figure 4.3: Hardware Setup II

Initially we used Setup I with the following configurations:

- Hardware has 2 TB Hard Disk and 32 GB RAM.
- Windows Server with 2012 Standard edition.
- Hyper-V Manager of Microsoft Corporation having Version: 6.2.9200.16384
- Ubuntu 16.04 LTS.

With Setup-I, we didn't get any significant progress. With this setup we couldn't get internet connectivity to the cloud (openstack) instance as shown in Fig 4.1. We tried to resolve this issue by using different methods like dedicating a whole new subnet to the Ubuntu Desktop and hence to the openstack instances, configuring the openvswitch (ovs) bridge manually in local.conf file while installing devstack on Ubuntu desktop having single NIC (Network Interface Card) and also on the one with two NICs, setting static ip routes using floating ips, trying to keep network manager of Ubuntu desktop out of the picture as we inferred

that it was the network manager which was interfering with openstack network configuration, etc. But none of this worked.
After getting torn up by Setup I, we adopted Setup II which is our current setup with following configurations:

- Hardware has 1 TB Hard Disk and 16 GB RAM.
- Ubuntu 16.04 LTS.

All those challenges that we faced with Setup I didn't occur with Setup II. We finally inferred that Hyper-V may be causing some issues but were unable to confirm the same or find the source of those issues.
In brief, Hyper-V technology provides an environment that you can use to create and manage virtual machines and their resources. Each virtual machine is an isolated, virtualized computer system that is capable of running its own operating system. This allows you to run multiple operating systems at the same time on the same physical computer. The type of virtualization offered by Hyper-V is known as hardware virtualization.
With Setup I, we tried to install openstack on Ubuntu 16.04 LTS which is a virtual machine created by Hyper-V manager. Virtualization offered by Hyper-V might not be in synchronization with virtualization that openstack was trying to offer at all. This might be the reason that instances were not able to connect to the internet. While in Setup II Ubuntu was not a VM so everything worked out as it was supposed to.
In the next chapter, we will discuss the analysis and implementation of our solution in an enterprise.

5

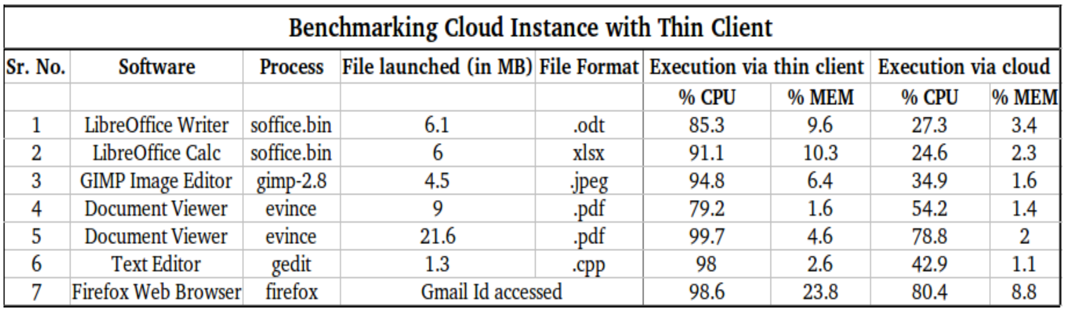
Analysis

Performance Comparison

We have can access the cloud instance having 4 GB of RAM and 50 GB of Disk size via Remote Desktop Protocal(RDP) in Raspberry Pi having 1 GB of RAM and 8 GB of disk size.

We compare the performance of thin-client Raspberry pi with the Ubuntu instance launched using openstack accessed by thin-client Raspberry Pi itself as described below.

We launch various applications that are prominently used by enterprises (as



Benchmarking Cloud Instance with Thin Client

Sr. No.	Software	Process	File launched (in MB)	File Format	Execution via thin client		Execution via cloud	
					% CPU	% MEM	% CPU	% MEM
1	LibreOffice Writer	soffice.bin	6.1	.odt	85.3	9.6	27.3	3.4
2	LibreOffice Calc	soffice.bin	6	xlsx	91.1	10.3	24.6	2.3
3	GIMP Image Editor	gimp-2.8	4.5	.jpeg	94.8	6.4	34.9	1.6
4	Document Viewer	evince	9	.pdf	79.2	1.6	54.2	1.4
5	Document Viewer	evince	21.6	.pdf	99.7	4.6	78.8	2
6	Text Editor	gedit	1.3	.cpp	98	2.6	42.9	1.1
7	Firefox Web Browser	firefox	Gmail Id accessed		98.6	23.8	80.4	8.8

Figure 5.1: Performance comparison between Raspberry Pi and Cloud Instance

shown in Fig 5.1) in the cloud instance as well as in the Raspberry Pi. We considered %CPU usage and %MEM usage as benchmarks. We gathered the data shown in Fig 5.1 by running top command at the time of launching the applications.

CPU Usage (%CPU) is the percentage of your CPU that is being used by the process. By default, top displays this as a percentage of a single CPU. On multi-core systems, you can have percentages that are greater than 100%. For

example, if 3 cores are at 60% use, top will show a CPU use of 180%.
Memory Usage (%MEM) : A task's currently used share of available physical memory (RAM).
All the seven applications have notably lesser consumption of resources (i.e. %CPU & %MEM) when launched in openstack instance via RDP in Raspberry Pi as compared to when launched in Raspberry Pi itself.
This implies that openstack instance has left with greater RAM after launching of applications. Instance with the more RAM has the larger digital countertop to work on and programs will run faster. It means openstack instance can launch more than one application almost simultaneously while Raspberry may not be able to do the same due to scarcity of resources after the first high-end application launched. Hence, cloud instance has better performance than Raspberry Pi.

Implementation in an Enterprise

A new employee joining the business enterprises having our cloud-based solution will be given a configured Raspberry Pi, Monitor, Keyboard, Mouse and Internet Connection along with an IP address (which would be the floating IP address), username and password of a cloud-based Ubuntu instance. The employee will also be provided with a url along with the username (which would be same as the username of cloud-based instance) and password for controlling the status of his instance by resuming or pausing. Resume functionality would dynamically allocate the resources (i.e. RAM) to the cloud instance and pause functionality would release the resources. In this way memory (RAM) can be dynamically allocated "On Demand".
Employees would be experiencing high-end applications (ie. applications that require significant amount of CPU% and MEM%) via a low budget thin client Raspberry Pi rather than a conventional high budget PC.
In the next chapter, we will discuss about the future scope of our work.

6

What's Next?

- Horizontal scaling of this system can be tested by attaching more number of servers.

- A complete functionality website that can launch instance and create user automatically.

References

[1] Cynthia Harvey. *60 Open Source Apps You Can Use in the Cloud.* https://www.datamation.com/open-source/60-open-source-apps-you-can-use-in-the-cloud-1.html. (visited on May 1, 2017)

[2] *Cloud Computing* https://en.wikipedia.org/wiki/Cloud_computing (visited on May 3, 2017)

[3] *What is Cloud computing?* https://azure.microsoft.com/en-in/overview/what-is-cloud-computing/ (visited on May 5, 2017)

[4] *OpenStack Docs : Getting started* https://developer.openstack.org/firstapp-libcloud/getting_started.html (visited on May 10, 2017)

[5] *OpenStack Architecture Design Guide* https://docs.openstack.org/arch-design/ (visited on May 13, 2017)

[6] *OpenStack : Basic Architecture* https://docs.openstack.org/glance/pike/contributor/architecture.html (visited on May 27, 2017)

[7] *OpenStack : Conceptual Architecture* https://docs.openstack.org/newton/install-guide-rdo/common/get-started-conceptual-architecture.html (visited on June 15, 2017)

[8] *Overcommitting CPU and RAM* https://docs.openstack.org/arch-design/design-compute/design-compute-overcommit.html (visited on July 8, 2017)

[9] Jack Wallen. *How to install OpenStack on a single Ubuntu Server virtual machine.* https://www.techrepublic.com/article/how-to-install-openstack-on-a-single-ubuntu-server-virtual-machine/ (visited on July 27, 2017)

[10] *All-In-One Single Machine* https://docs.openstack.org/devstack/latest/guides/single-machine.html (visited on Aug 5, 2017)

[11] *Devstack* https://docs.openstack.org/devstack/latest/ (visited on August 11, 2017)

[12] *OpenStack in 20 minutes* http://therandomsecurityguy.com/openstack-20-minutes/ (visited on August 25, 2017)

[13] *openstack-dev/devstack* https://github.com/openstack-dev/devstack/blob/master/stack.sh (visited on September 1, 2017)

[14] Vakhtang Z. *Stack.sh error (./stack.sh: line 463: generate-subunit: command not found)* https://linuxacademy.com/community/posts/show/topic/9058-stacksh-error-stacksh-line-463-generatesubunit-command-not-found (visited on September 6, 2017)

[15] *Get images* https://docs.openstack.org/image-guide/obtain-images.html (visited on September 11, 2017)

[16] *Ubuntu Cloud Images* https://cloud-images.ubuntu.com/ (visited on September 13, 2017)

[17] Udara S. S Liyanage. *Cannot create volume of more than 2 GBs in openstack.* https://stackoverflow.com/questions/15835215/cannot-create-volume-of-more-than-2-gbs-in-openstack (visited on September 22, 2017)

[18] *Using DevStack with neutron Networking* https://docs.openstack.org/devstack/latest/guides/neutron.html (visited on October 2, 2017)

[19] *[OpenStack]: Set user password when launching cloud images* `https://techglimpse.com/nova-boot-instance-with-password/` (visited on October 5, 2017)

[20] *Allowing guests in DevStack to talk to outside world* `https://ask.openstack.org/en/question/1830/allowing-guests-in-devstack-to-talk-to-outside-world/` (visited on October 12, 2017)

[21] *OpenStack Networking* `https://docs.openstack.org/mitaka/networking-guide/intro-os-networking.html` (visited on October 18, 2017)

[22] *Linux: Iptables List and Show All NAT IPTables Rules Command* `https://www.cyberciti.biz/faq/howto-iptables-show-nat-rules/` (visited on October 20, 2017)

[23] Damian Igbe. *Identifying and Troubleshooting Neutron Namespaces* `https://www.mirantis.com/blog/identifying-and-troubleshooting-neutron-namespaces/` (visited on October 24, 2017)

[24] *Introducing Linux Network Namespaces* `https://blog.scottlowe.org/2013/09/04/introducing-linux-network-namespaces/` (visited on November 7, 2017)

[25] Dash Blinken Lichten. *Dash Blinken Lichten-An introduction to network namespaces* `http://www.dasblinkenlichten.com/an-introduction-to-network-namespaces/` (visited on November 10, 2017)

[26] *Network address translation* `https://docs..org/mitaka/networking-guide/intro-nat.html` (visited on November 11, 2017)

[27] Griffon. *Ubuntu 14.04 – How to install xrdp in Ubuntu 14.04* `http://c-nergy.be/blog/?p=5305` (visited on November 12, 2017)

[28] *Django documentation* `https://docs.djangoproject.com/en/1.11/` (visited on November 15, 2017)

[29] *SSH (Secure Shell) : Raspberry Pi Configuration* `https://www.raspberrypi.org/documentation/remote-access/ssh/` (visited on November 17, 2017)

[30] *3 Ways to Run a Remote Desktop on Raspberry Pi* `https://eltechs.com/3-ways-to-run-a-remote-desktop-on-raspberry-pi/` (visited on November 17, 2017)

[31] Alex Chamberlain. *Does Raspbian come with a remote desktop (RDP) client?* `https://raspberrypi.stackexchange.com/questions/2130/does-raspbian-come-with-a-remote-desktop-rdp-client` (visited on November 18, 2017)

[32] *Resources-Documentation — Pycharm* `https://www.jetbrains.com/pycharm/documentation/` (visited on November 19, 2017)

[33] *Getting started with the SDK* `https://developer.openstack.org/sdks/python/openstacksdk/users/index.html` (visited on November 20, 2017)